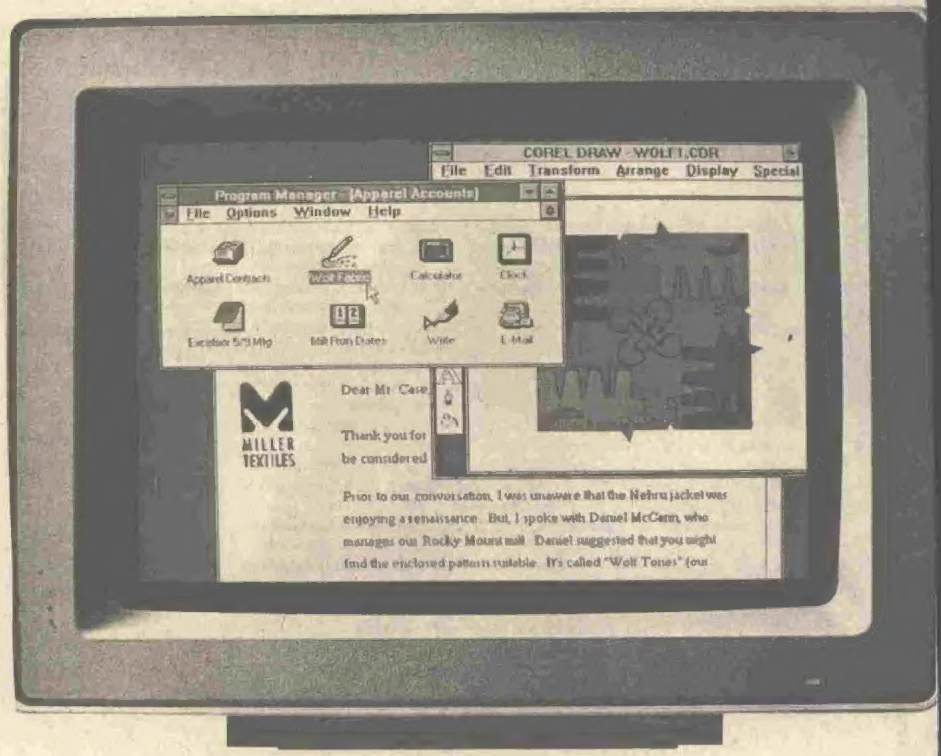
บุญเลิศ เอี่ยมทัศนา

ซอฟต์แวร์ชิพ: บานหน้าต่าง

การประกอบชิพ (สกุล) ที่แยกกันสร้างเข้าด้วยกัน และทำการแบ่งหน้าต่างออกเป็นหลายบาน



บทความในชุดซอฟต์แวร์ชิพ เนิ่นนานมาพอสมควร ทั้งนี้เพราะผมต้องห่างเครื่องเป็นระยะ ๆ มา พอถึงตอนที่จะจับมาต่อคือประกอบกันก็เห็นว่า ทางซีกหน้าต่างเขียนไว้เดิมไม่ดีเท่าที่ควร ประกอบกับผมต้องจับงานที่ค้างมานานเสียที ในงานใหม่นี้ต้องการหน้าต่างที่มีบานหน้าต่างได้หลายบาน ผมจึงขอปรับปรุงโปรแกรมทางซีกหน้าต่างและจับงานที่ค้างก่อน

การประกอบชิพ

แต่สำหรับท่านที่ติดตามบทความชุดนี้มานาน คงต้องการทราบว่าจะประกอบชิพ (สกุล) ที่แยกกันสร้างเข้าด้วยกันได้อย่างไร เพื่อไม่ต้องรอนาน ผมขอเรียนหลักการให้ทราบก่อน กล่าวคือ

เมื่อสกุลหนึ่งต้องการใช้บริการจากภายนอก สามารถดำเนินการได้โดยกำหนดกิจกรรมให้เป็นเวอร์ชวล แล้วไปเรียกใช้บริการในสกุลลูก เช่น ในเท็กซ์-อีดิตต้องการให้ผู้ใช้ป้อนชื่อไฟล์ ซึ่งต้องดำเนินการผ่านจอภาพ แต่ขณะที่เขียนเท็กซ์อีดิตนั้นไม่ทราบว่าจะเป็นจอภาพแบบใด ตามที่ได้ออกแบบไว้ว่าให้วินโดว์อีดิตดำเนินการเกี่ยวกับจอภาพ

แต่ถ้าอีกสกุลหนึ่งต้องการใช้บริการจากภายนอกเช่นกัน ดังเช่นวินโดว์อีดิตต้องการแจ้งความสูงของหน้าต่างใหม่ให้เท็กซ์อีดิตทราบ จะใช้วิธีกำหนดกิจกรรมให้เป็นเวอร์ชวลไม่ได้ เพราะในเทอร์โบปาสคาล (รุ่น 5.5 และ 6.0) ให้สืบสกุลได้จากบรรพบุรุษเดียวเท่านั้น

ทางออกคือ ใช้ตัวแปรโปรแกรมย่อย ทำให้สามารถเรียกใช้โปรแกรมย่อยใดก็ได้ที่มีส่วนหัวเหมือนกับที่กำหนดให้แก่ตัวแปรได้ล่วงหน้า แต่จะเรียกโปรแกรมย่อยที่เป็นกิจกรรมในสกุลอื่นไม่ได้ต้องต่อเชื่อมให้

ขอให้ดูตัวอย่างดีกว่าครับตามรูปที่ 1 สมมติว่าสกุล TextEd เป็นเท็กซ์อีดิตดังกล่าวมา ต้องการใช้บริการทางด้านเอาต์พุตของวินโดว์อีดิต จึงได้กำหนดกิจกรรม OutStr ไว้เป็นเวอร์ชวล และได้เขียนโปรแกรมย่อยของกิจกรรมนี้ไว้ด้วย เพื่อใช้ในขณะที่เขียนโปรแกรมอื่น ๆ ในสกุลนี้ ที่เรียกใช้กิจกรรมนี้ เช่น Action กับได้กำหนดกิจกรรม ChngWndHi ไว้ด้วยเพื่อเปลี่ยนความสูงของหน้าต่าง ซึ่งในโปรแกรมจริงจะต้องดำเนินการในเรื่องอื่น ๆ อีกมาก

สำหรับสกุลวินโดว์อีดิตสมมติคือ ScrnEd อยู่ในรูปที่ 2 ขอให้ดูการ

```
PROGRAM  DemoConnectChips;

(* ==============  UNIT  TEXT  ========================== *)

TYPE
  TextEd = OBJECT
    tstr : String;
    thi  : Integer;

  constructor  Init(nstr : String);
  procedure  OutStr;  virtual;
  procedure  ChngWndHi(nhi : Integer);
  procedure  Action;
  destructor  Fin;
          END;


CONSTRUCTOR  TextEd.Init(nstr : String);
BEGIN  tstr := nstr;  thi := 0    END;


PROCEDURE  TextEd.OutStr;
BEGIN  WriteLn('TextEd: ',tstr,thi)   END;


PROCEDURE  TextEd.ChngWndHi(nhi : Integer);
BEGIN  thi := nhi   END;


PROCEDURE  TextEd.Action;
BEGIN   OutStr       END;


DESTRUCTOR  TextEd.Fin;
BEGIN       END;
```

รูปที่ 1

```
TYPE
  WndHiProc = procedure(nhi : Integer);

  ScrnEd  = OBJECT
    whi    : Integer;
    callChngHi : WndHiProc;

  constructor  Init(nhi : Integer);
  procedure  ChngHi;
  procedure  OutputStr(ostr : String;num : Integer );
  destructor  Fin;
           END;

{$F+}
procedure  DummyHi(nhi : Integer);
begin                          end;
{$F-}


CONSTRUCTOR  ScrnEd.Init(nhi : Integer);
BEGIN  whi := nhi;  callChngHi := DummyHi  END;


PROCEDURE  ScrnEd.ChngHi;
BEGIN  inc(whi);  callChngHi(whi)  END;


PROCEDURE  ScrnEd.OutputStr(ostr : String; num : Integer );
BEGIN  WriteLn('ScrnEd: ',ostr,num);  ChngHi  END;


DESTRUCTOR  ScrnEd.Fin;
BEGIN       END;
```

รูปที่ 2

กำหนดแบบข้อมูล WndHiProc ว่าตรงกับ ChngWndHi ในสกุล TextEd กับการกำหนดตัวแปรโปรแกรมย่อย CallChngHi กิจกรรมที่ยกมาเป็นตัวอย่างคือ ChngHi ซึ่งจะดำเนินการต่าง ๆ เมื่อผู้ใช้เปลี่ยนขนาดของหน้าต่าง ซึ่งในตัวอย่างเพียงแต่ให้เพิ่มความสูงของหน้าต่างขึ้นหนึ่ง

ส่วน OutPutStr คือกิจกรรมที่จะให้บริการในการนำสตริงไปออกจอภาพ แต่เพื่อแสดงการทำงานของ ChngHi จึงให้เรียกใช้ ChngHi ด้วย

เมื่อเขียนโปรแกรมของแต่ละสกุลเสร็จแล้วก็มาถึงขั้นตอนที่จะนำมาประกอบเข้าด้วยกันดังรูปที่ 3 ได้กำหนดสกุล Editor เป็นผู้สืบสกุลของ TextEd ทั้งนี้เพราะมีกิจกรรม OutStr ที่จะต้องดำเนินการต่อ และได้กำหนด screen ให้เป็นอินสแตนส์ (ตัวแปร) ของสกุล ScrnEd (ขอให้สังเกตตรงนี้ด้วยครับ เพราะถ้าเป็นการรวมสกุลที่ไม่มีความสัมพันธ์กัน อาจกำหนดอินสแตนซ์ของสกุล TextEd ได้เช่นกัน)

ขอกล่าวทางด้าน TextEd ก่อน ที่ต้องการใช้บริการของ ScrnEd จะปรากฏในโปรแกรมย่อยของ OutStr (รูปที่ 3) คือที่ส่งสื่อความ OutputStr ให้แก่ screen

ส่วนทางด้าน ScrnEd ที่จะต่อ callChngHi เข้ากับโปรแกรมย่อยที่จะดำเนินการในเรื่องนี้ อยู่ในรูปที่ 4 (โปรแกรมในรูปนี้คือ โปรแกรมใช้งานที่แท้จริง ซึ่งในการดำเนินการจริง ควรเขียนโปรแกรมตามรูปที่ 3 ไว้ในยูนิต) เริ่มจากการกำหนดตัวแปร note ให้เป็นอินสแตนซ์ของ Editor แล้วเขียนโปรแกรมย่อย TranHi ให้มีส่วนหัวแบบเดียวกับ callChngHi ในโปรแกรมนี้ก็เพียงแต่นำค่าที่ผ่านมา ให้แก่ Chng-

```
(* ================= COMBINE CLASS ====================== *)

TYPE
    Editor = OBJECT(TextEd)
     screen : ScrnEd;

  constructor  Init(astr : String; wndhi : Integer);
  procedure  OutStr;  virtual;
  destructor  Fin;
              END;


CONSTRUCTOR  Editor.Init(astr : String; wndhi : Integer);
BEGIN
  TextEd.Init(astr);  screen.Init(wndhi);
END;

PROCEDURE  Editor.OutStr;
BEGIN  screen.OutputStr(tstr,thi);  END;

DESTRUCTOR  Editor.Fin;
BEGIN   screen.Fin;  TextEd.Fin;   END;
```

รูปที่ 3

```
(* ================= TEST  PROGRAM ====================== *)

VAR   note : Editor;

{$F+}
procedure  TranHi(nhi : Integer);
begin  note.ChngWndHi(nhi)   end;
{$F-}


BEGIN   { MAIN }
  WriteLn;
  note.Init('Window hight = ',15);
  note.screen.callChngHi := Tranhi;
  note.Action;  note.Action;  note.Action;
  note.Fin;
END.
```

รูปที่ 4

WndHi ของ note

ซึ่งในโปรแกรมหลักจะต่อ callChngHi เข้ากับ Tranhi ดังนั้นเมื่อส่งสื่อความ Action ให้แก่ note จะเป็นการเรียกใช้ OutStr ใน Editor (ตามหลักของเวอร์ชวล) ซึ่งจะเรียกใช้ OutputStr ใน ScrnEd อีกทอดหนึ่ง นำข้อความที่ให้ไว้เมื่อ Init ออกจอภาพ (บรรทัดแรกของรูปที่ 5)

เสร็จแล้วจะเรียกใช้ ChngHi ซึ่งจะเพิ่มค่า whi ขึ้นหนึ่ง แล้วผ่านค่านี้ไปให้แก่ตัวแปรโปรแกรมย่อย callChngHi ซึ่งต่ออยู่กับและเท่ากับเป็นการผ่านค่าไปให้ TranHi ซึ่งจะผ่านค่าไปให้แก่ ChngWndHi ของ note ใน TextEd อีกทอดหนึ่ง เพื่อให้ค่าแก่ตัวแปร thi ดังนั้นเมื่อเราใช้สื่อความ Action ครั้งแรก ค่า thi จึงเป็นศูนย์ตาม Init ของ

```
ScrnEd: Window hight = 0
ScrnEd: Window hight = 16
ScrnEd: Window hight = 17
```

รูปที่ 5 ผลการรันโปรแกรมตามรูปที่ 4

TextEd ครั้งที่สองจึงเป็น 15 บวก 1

วิธีการนี้โปรแกรมทำงานได้ถูกต้อง แต่สกุล Editor ไม่สมบูรณ์ในตัวเอง ไม่ถูกต้องตามหลักการเขียนโปรแกรมในแบบโอโอพี ในบทความชุดซอฟต์แวร์ชิพที่ผ่านมาจึงเปลี่ยนเป็นให้สกุลวินโดว์ แจ้งให้สกุลเอดิเตอร์ทราบทางรหัสคำสั่ง แล้วให้กิจกรรมในเอดิเตอร์เรียกใช้กิจกรรมในวินโดว์ (เช่น ChngHi) ดำเนินการโดยตรง

การประกอบชิพเข้าด้วยกันโดยหลักใหญ่มีเพียงเท่านี้ครับ คิดว่าท่านที่ต้องการเขียนโปรแกรมแยกกันเป็นชิพหรือเป็นโมดูล ๆ เพื่อนำมาประกอบกันคงได้ประโยชน์จากบทความนี้ ส่วนบทความในชุดซอฟต์แวร์ชิพที่เมื่อประกอบกันแล้วจะได้เป็นเอดิเตอร์นั้น คงต้องรอก่อนครับเพราะผมต้องการได้เอดิเตอร์ที่ใช้งานได้จริง ๆ ซึ่งต้องเริ่มจากการปรับปรุงสกุลหน้าต่างและต้องต่อไปงานอื่นก่อน สำหรับสกุลหน้าต่างใหม่นี้ผมได้นำมาเสนอด้วย เพราะสามารถนำไปใช้งานอื่นได้

หน้าต่างที่มีหลายบาน

ในโครงงานใหม่ที่ผมจะดำเนินการต่อไปอีกซีกหนึ่งนั้น ต้องการหน้าต่างที่สามารถแบ่งออกเป็นบานหน้าต่างหลายบานได้ ดังรูปที่ 6 เมื่อผมต้องปรับปรุงสกุลหน้าต่างจึงถือโอกาสให้มีขีดความสามารถนี้ด้วย จึงมีเรื่องใหม่ที่ขอนำมาเล่าสู่กันฟังเพื่อให้ใช้งานสกุลนี้ได้ คือ

ตัวแปร

มีการกำหนดตัวแปรเพิ่มขึ้นดังแสดงในรูปที่ 7 ซึ่งแสดงแบบข้อมูล WndParmt กับตอนต้นของการกำหนดสกุล SimWnd (ย่อมาจาก Simple-Window เพื่อไม่ให้ชื่อซ้ำกับของเดิมคือ NormalWnd) มีตัวแปร ดังนี้

- wparm เป็นตัวแปรสำหรับเก็บค่าพารามิเตอร์ของหน้าต่างที่ผู้เขียนโปรแกรมคนต่อไปจะส่งมาให้ ได้แก่ (ขอให้ดูการกำหนดแบบข้อมูล WndParmt ประกอบ) ชื่อหน้าต่าง (mame) บานหน้าต่าง (pale) ซึ่งจะเป็นบานหลักคือเป็นบานใหญ่ทั้งหน้าต่าง เป็นอะเรย์ขนาด 6 ช่อง เก็บค่าต่อไปนี้ตามลำดับ คือ โคออร์ดิเนตของจุดมุมบนซ้ายในแนวแกน x และ y ความกว้าง ความสูง ตำแหน่งของเคอร์เซอร์ในแนวแกน x และ y รายละเอียดต่อไปคือ horline กับ verline เป็นอะเรย์ขนาด 10 ช่อง เพื่อเป็นค่า row และ column สำหรับลากเส้นขวางหน้าต่างในแนวนอนและแนวดิ่งตามลำดับ ใช้ทำกรอบของบานหน้าต่างด้วย

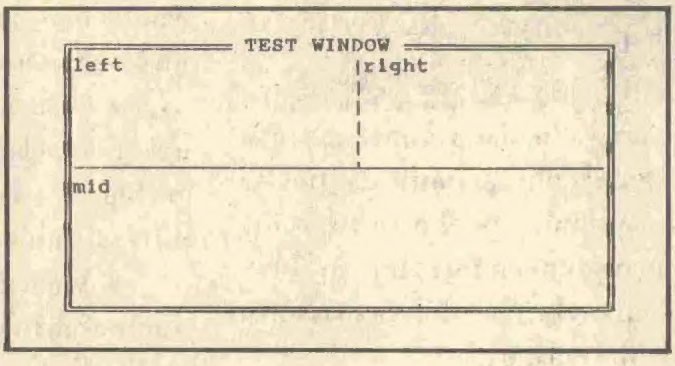
- subpale หากหน้าต่างมีหลายบาน จะต้องให้ค่าพารามิเตอร์ของบานหน้าต่างที่จะเปิดใช้แก่ตัวแปรนี้ (มีกิจกรรมดำเนินการให้) มีค่าต่าง ๆ เช่นเดียวกับบานหลัก จะแตกต่างกันตรงการอ้างอิงจุดมุมบนซ้าย ของบานหลักอ้างถึงจอภาพ แต่ของบานรองอ้างถึงบานหลักอีกทอดหนึ่ง ทั้งนี้เพื่อความสะดวกในการเคลื่อนย้ายหน้าต่างและเป็นการอำนวยความสะดวกแก่ผู้เขียนโปแกรมคนต่อไปด้วย

- curpale เก็บค่าของ subpale ที่เปลี่ยนค่าของจุดมุมบนซ้ายให้เป็นอ้างถึงจอภาพ หรือ pale ของบานหลักในเมื่อมีเพียงบานเดียวหรือเมื่อเคลื่อนย้ายและย่อขยายขนาดของหน้าต่าง

- wx1 กับ wy1 และ wx2 กับ wy2 คือ โคออร์ดิเนตของจุดมุมบนซ้ายและล่างขวาตามลำดับ ของบานหน้าต่างที่เปิดใช้ขณะนั้น (ความจริงไม่ต้องมีตัวแปรเหล่านี้เพราะมีอยู่แล้วใน curpale กำหนดขึ้นเพื่อความสะดวกในการเขียนโปรแกรมเท่านั้น รวมทั้งตัวแปรที่จะกล่าวต่อไปนี้ เว้น foregrnd กับพวกแฟลกต่าง ๆ)

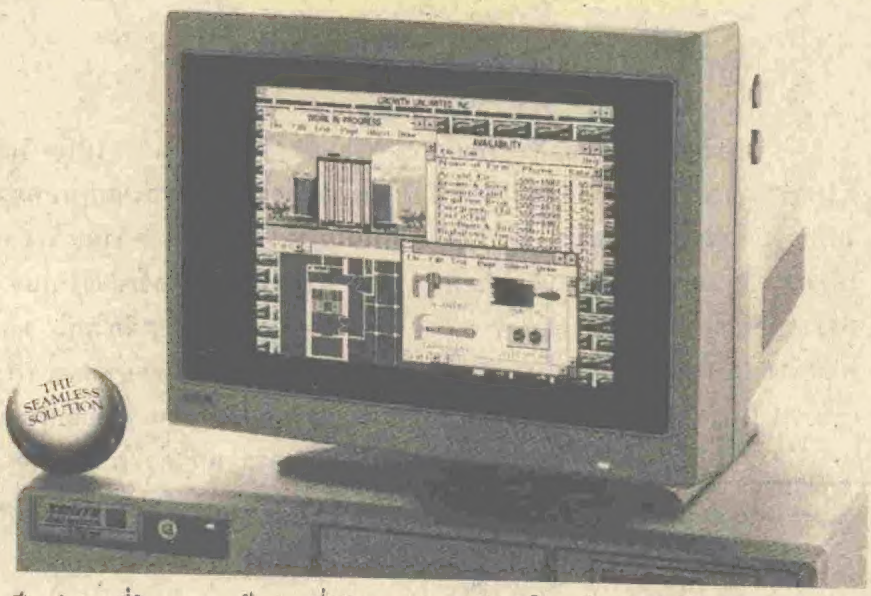
- curx กับ cury คือตำบลที่ของเคอร์เซอร์ในแนวแกน x และ y ตาม

รูปที่ 6
แสดงหน้าต่างที่มีบานหน้าต่าง 3 บาน




```
UNIT  SMPLWND;

Interface

uses  crt,prime;

CONST  MAXWID = 80;  MAXHI = 25;  NONE = 127;

TYPE

  OneColT = RECORD ch : Char; attr : Byte  END;
  OneRowT = ARRAY[1..80] OF OneColT;
  ScreenT = ARRAY[1..25] OF OneRowT;
  ScreenPtrT = ^ScreenT;
  RowPtrT = ^OneRowT;
  AryShrt = ARRAY[1..10] OF ShortInt;
  PtrAryShrt = ^AryShrt;
  PaleT  = ARRAY[1..6] OF Integer;

  WndParmT = RECORD
    name  : Str49;
    pale : PaleT;
    horline,verline : AryShrt;
            END;

VAR   screen : ScreenT ABSOLUTE $B000:$0000;
      backscrn : ScreenT;
{      screen : ScreenT ABSOLUTE $B800:$0000;  }
      lnebuf : OneRowT;


TYPE
  PtrSimWnd = ^SimWnd;
  SimWnd = OBJECT
    wparm : WndParmT;
    curpale,subpale : PaleT;
    wx1,wx2,wy1,wy2,curx,cury : Integer;
    xx1,xx2,yy1,yy2 : Integer;
    wndwid,wndhi,refx,refy,rgtlmt : Integer;
    wndbuffsi       : Word;
    foregrnd        : RowPtrT;
    snglpale,insflg,alphakey : Boolean;
```

รูปที่ 7



ลำดับเป็นตำบลที่ในบานหน้าต่างที่เปิดใช้ขณะนั้น

- xx1 กับ yy1 และ xx2 กับ yy2 เก็บโคออร์ดิเนตของจุดมุมบนซ้ายและมุมล่างขวา ของกรอบหน้าต่าง

- wndwid และ wndhi คือความกว้างและความสูงของบานหน้าต่างขณะนั้น

- refx และ refy คือค่า wx-1 และ wy-1 ตามลำดับ

- foregrnd เพื่อเก็บภาพหน้าต่างรวมกรอบ เมื่อปิดหน้าต่างด้วย Hide

- wndbuffsi เก็บขนาดของหน้าต่างรวมกรอบคือ ขนาดของหน่วยความจำในฮีพของ foregrnd

— snglpale เมื่อเป็นจริงแสดงว่าเป็นหน้าต่างที่มีบานหน้าต่างเพียงบานเดียว

- insflg เมื่อเป็นจริงให้แสดงภาพอักขระบนหน้าต่างด้วยวิธีแทรก หากเป็นเท็จจะเขียนทับ

- alphakey เมื่อเป็นจริงแสดงว่าผู้ใช้ได้กดคีย์ที่ไม่ใช่คีย์ควบคุมหรือคีย์หน้าที่พิเศษ

ช่องบรรทัด

เมื่อกำหนดหน้าต่างและบานหน้าต่าง ความสูงของบานหน้าต่างคือจำนวนบรรทัด ในแต่ละบรรทัดจะแสดงภาพอักขระได้เป็นจำนวนเท่ากับความกว้างของบานหน้าต่าง แต่ในบางกรณีต้องการแบ่งบรรทัดออกเป็นช่วง ๆ อีก จึงขอเรียกช่องในบรรทัดนี้ว่าช่องบรรทัดในการกล่าวถึงต่อไป

การกำหนดช่องบรรทัดทำได้หลายวิธี วิธีหนึ่งคือให้ wx1 เป็นเขตทางซ้ายและ rgtlmt เป็นเขตทางขวา

กิจกรรมพื้นฐาน

การกำหนดกิจกรรมในสกุล SimWnd ใกล้เคียงกับของเดิมมาก (เพราะเป็นการปรับปรุง) แต่เพื่อเป็นการทบทวนและไม่ต้องย้อนกลับไปค้นตอนเก่า ๆ ผมจะขอนำมากล่าวถึงใหม่อย่างสั้น ๆ สำหรับกิจกรรมที่เป็นพื้นฐานของหน้าต่างตามรูปที่ 8 ได้แก่

● Init เตรียมค่าตามพารามิเตอร์ที่ผ่านมาทาง wndparm ทำให้เป็นหน้าต่างที่มีบานหน้าต่างเพียงบานเดียว

● Show เปิดหน้าต่างพร้อมทั้งแสดงภาพหน้าต่าง ถ้าเป็นครั้งแรกจะตีกรอบและเส้นคั่น ถ้าเป็นครั้งต่อ ๆ ไปจะนำภาพหน้าต่างที่เก็บไว้ออกแสดง (เรียกใช้ WndOn)

● Hide ปิดหน้าต่างในแบบเก็บภาพหน้าต่าง คือเก็บภาพหน้าต่างไว้กับตัวแปร foregrnd และเก็บคือลบภาพหน้าต่างบนจอภาพ (เรียกใช้ WndOff)

● WndOn เปิดใช้หน้าต่างที่มีภาพหน้าต่างอยู่แล้วบนจอภาพ ให้เคอร์เซอร์อยู่ ณ ตำแหน่งที่เก็บไว้ในบานหน้าต่าง (curpale)

● WndOff ปิดหน้าต่างในแบบคงภาพหน้าต่างใช้บนจอภาพ เก็บตำแหน่งเคอร์เซอร์ไว้ในบานหน้าต่าง (curpale)

● OpenSubWnd เปิดใช้บานหน้าต่างที่ผ่านค่ามาทาง pale (สามารถเปิดปิดหน้าต่างตามที่กล่าวมา เพื่อไปเปิดหน้าต่างอื่นได้ และเมื่อกลับมาจะกลับมาที่บานหน้าต่างนี้)

● CloseSubWnd ปิดบานหน้าต่างขณะนั้น (เพื่อเปิดใช้บานอื่น) โดยเก็บตำแหน่งของเคอร์เซอร์ให้แก่ pale เท่านั้น (ดังนั้นหากไม่ต้องการตำแหน่งนี้ ไม่ต้องปิด)

● MoveTo ย้ายหน้าต่างไปยังจุด nx1, ny1 (จุดโคออร์ดิเนตของมุมบนซ้ายของหน้าต่าง ณ ตำแหน่งใหม่)

```
constructor  Init(wndparm : WndParmT);
procedure    Hide; virtual;
procedure    Show; virtual;
procedure    WndOn;
procedure    WndOff;
procedure    OpenSubWnd(pale : PaleT);
procedure    CloseSubWnd(var pale : PaleT);

procedure    MoveTo(nx1,ny1 : Integer);
procedure    ResizeWnd(nx1,ny1,nwid,nhi : Integer);
procedure    Drag;
procedure    ExpCond;
```

รูปที่ 8

● ResizeWnd ปรับขนาดของหน้าต่างใหม่เป็น nx1, ny1 คือจุดโคออร์ดิเนตของมุมบนซ้าย nwid และ nhi คือความกว้างและความสูงของหน้าต่างตามลำดับ (ต้องเก็บหน้าต่างก่อน)

● Drag เลื่อนหน้าต่างตามความต้องการของผู้ใช้ด้วยคีย์ลูกศร (คีย์ลูกศรทั้งสี่และคีย์ Home, PgUp, PgDn กับ End ในทิศทางจากคีย์เลข 5 ออกไป) จนกว่าจะพอใจให้ยุติด้วยการกดคีย์ Enter การย้ายจะก้าวครั้งละ 1 โรว์หรือคอลัมน์ (แล้วแต่ทิศทาง) หากต้องการย้ายแบบกระโดดให้กดคีย์ 2 ถึง 0 (0 เท่ากับ 10) จะกระโดดครั้งละ 4 ถึง 20 ก้าว (เปลี่ยนค่ากระโดดใหม่ได้)

● ExpCond ให้ผู้ใช้ขยายและย่อขนาดของหน้าต่างด้วยคีย์ลูกศร (และคีย์อื่นดังกล่าวมา) เริ่มด้วยการขยายคือจะขยายหน้าต่างลูกศรออกไปในทิศทางของลูกศร หากผู้ใช้ต้องการย่อให้กดคีย์ ins จะย่อหน้าต่างด้านลูกศรในทิศทางตรงข้าม หากกดคีย์ ins ซ้ำจะกลับไปขยายใหม่ (เรียกใช้ ResizeWnd)

กิจกรรมในการลากเส้น

กิจกรรมในส่วนนี้ตามรูปที่ 9 กำหนดขึ้นเมื่อการตีกรอบหน้าต่าง แต่เห็นควรให้เป็นกิจกรรมสำหรับใช้งานทั่วไปด้วย (ตำแหน่งต่าง ๆ ที่จะกล่าวถึงในหัวข้อนี้อ้างถึงตำบลที่ในบานหน้าต่าง ได้แก่)

● DrwaHar ลากเส้นในแนวนอน ณ โรว์ที่ row จากคอลัมน์ที่ frm ถึง too ด้วยอักขระ ch (หากให้ ch เป็น space จะเป็นการลบ)

● DrawVer ลากเส้นในแนวดิ่ง ณ คอลัมน์ที่ col จากโรว์ที่ frm ถึง too ด้วยอักขระ ch

```
procedure  DrawHor(row,frm,too : Integer; ch : Char);
procedure  DrawVer(col,frm,too : Integer; ch : Char);
procedure  PaintHor(row,frm,too : Integer; attri : Byte);
procedure  PaintVer(col,frm,too : Integer; attri : Byte);
procedure  OutCntrStr(ostr : String; row : Integer);
procedure  GetPict(var buf : RowPtrT);
procedure  PutPict(buf : RowPtrT);
```

รูปที่ 9

```
procedure  InsOnOff;
procedure  CurAtXY(newx,newy : Integer);
procedure  OutCh(ch : Char);
procedure  OutChAt(ch : Char; row,col : Integer);
procedure  OutStr(ostr : String; stt : Byte);
procedure  OutLnStr(ostr : String; stt : Byte);
procedure  OutStrAt(ostr : String;
                row,frm,wid : Integer; stt : Byte);
procedure  OutStrCont(ostr : String;
                row,frm,wid : Integer; var stt : Byte);
```

รูปที่ 10

● PaintHor เช่นเดียวกับ DrwaHor แต่เป็นการให้ค่าแอตทริบิวต์ (attribute : ทำให้เกิดเป็นภาพกลับเน้นและกะพริบ เป็นต้น) ตามที่จะให้แก่ attri

● PaintVer เช่นเดียวกับ DrawVer แต่เป็นการให้ค่าแอตทริบิวต์ใน attri

● OutCntrStr นำข้อความใน ostr ไปแสดง ณ กึ่งกลางบรรทัดที่ row (เรียกใช้ OutStrAt)

● GetPict เก็บภาพหน้าต่างรวมกรอบให้แก่ตัวแปรที่ผ่านค่ามาให้แก่ buf ซึ่งถ้ายังไม่กำหนดหน่วยความจำในฮีพจะดำเนินการให้ (อ้างถึง xx1, xx2, yy1 และ yy2 ขณะนั้น)

● PutPict นำภาพที่เก็บในตัวแปร buf ออกแสดงโดยอ้างถึง xx1, xx2, yy1 และ yy2 ขณะนั้น ซึ่งอาจเป็นการย้ายตำบลที่

กิจกรรมด้านเอาต์พุต

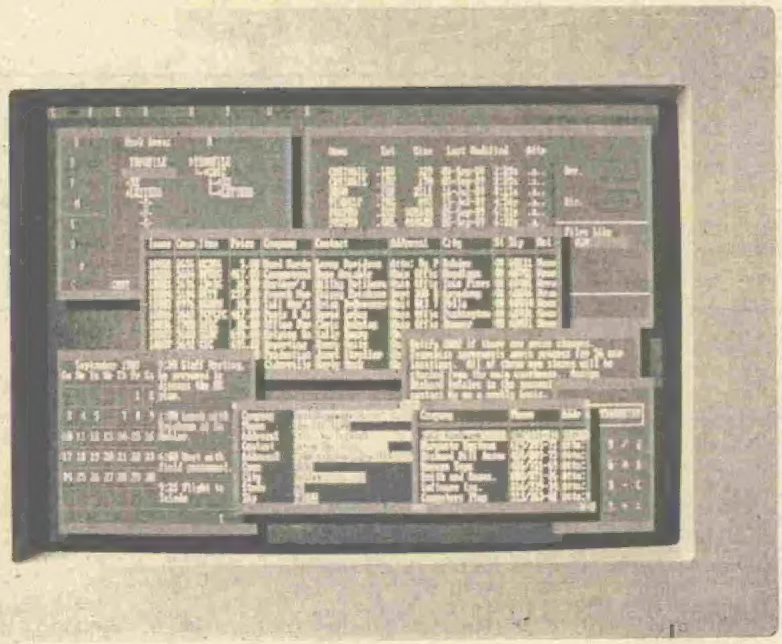
เพื่อเอาต์พุตค่าเป็นอักขระหรือสตริงกับจอภาพโดยตรง ซึ่งจะดำเนินการได้เร็วกว่าใช้ Write ดังรูปที่ 10 ได้แก่

● InsOnOff เพื่อกลับแฟล็ก insflg ให้เป็นจริงและเท็จสลับกันตามความต้องการของผู้ใช้

● CurAtXY เพื่อให้ค่า curx และ cury เป็น newx และ newy ตามลำดับ และให้เคอร์เซอร์อยู่ที่ตำแหน่งนี้ (ให้ใช้คำสั่ง CurAtXY แทน GotoXY นอกจากจะแน่ใจว่าไม่เกี่ยวข้องกับค่า curx และ cury)

● OutChAt นำภาพอักขระใน ch ไปออก ณ ตำแหน่งโรว์ row และคอลัมน์ col โดยตรวจ insflg ถ้าเป็นจริงจะนำออกแสดงด้วยวิธีแทรกภาพภายในช่องบรรทัด ถ้าเป็นเท็จจะนำออกแสดงโดยทับภาพอักขระเดิม

● OutCh นำภาพอักขระใน ch ไปออก ณ ตำแหน่งของเคอร์เซอร์ และ



เลื่อนเคอร์เซอร์กับ curx ไปทางขวา 1 คอลัมน์ (เรียกใช้ OutChAt)

● OutStrCont นำภาพอักขระในสตริง ostr ไปออก ณ โรว์ row ภายในช่องบรรทัดจากคอลัมน์ที่ frm เป็นจำนวน wid อักขระ โดยเริ่มจากอักขระที่ stt ในสตริง นั่นคือถ้าสตริงจาก stt ไปยาวกว่า wid ก็จะนำออกเพียง wid ตัว เสร็จแล้วจะให้ stt มีค่าเท่ากับตำแหน่งของอักขระในสตริงตัวสุดท้ายที่นำออกจอภาพ เพื่อการเอาต์พุตต่อเนื่องเช่นเมื่อผู้ใช้ต้องการอ่านสตริงทั้งหมด (ครั้งละช่องบรรทัด)

● OutStrAt - นำภาพอักขระในสตริงไปออกในบานหน้าต่าง เช่นเดียวกันและเรียกใช้ OutStrCont โดยเปิดโอกาสให้ผ่านค่าคงตัวมาให้แก่ stt ได้เท่านั้น

● OutStr นำภาพอักขระในสตริง ostr จากอักขระที่ stt ไปออกจอภาพ ณ ตำแหน่งของเคอร์เซอร์ โดยมีช่องบรรทัดเต็มบรรทัด และให้เคอร์เซอร์อยู่หลังอักขระตัวสุดท้ายที่นำออกแสดง

● OutLnStr เช่นเดียวกันและเรียกใช้ OutStr แต่ให้เคอร์เซอร์ (รวมทั้งค่า curx และ cury) ขึ้นบรรทัดใหม่ หากสตริงสั้นกว่าช่องบรรทัดจะลบภาพในช่องบรรทัดที่เหลือ

กิจกรรมด้านอินพุต

เพื่อการอินพุตอักขระหรือสตริงที่ต้องการให้อยู่ในควบคุมมากกว่าใช้ Read ดังรูปที่ 11 ได้แก่

● ReadChChf อ่านค่าคีย์โดยเรียกใช้ GetChChf (อยู่ในยูนิต prime) ได้ค่าคีย์ธรรมดารวมทั้งคีย์คอนโทรลใน ch หรือคีย์หน้าที่พิเศษใน chf นำ ch มาตรวจหากไม่ใช่คีย์คอนโทรลให้นำออกแสดง ณ ตำแหน่งเคอร์เซอร์ด้วย Outch

● DelCh ลบอักขระตรงตำแหน่งเคอร์เซอร์ 1 อักขระ โดยเลื่อนอักขระที่อยู่หลังเคอร์เซอร์ไปจนสุดช่องบรรทัดมาแทนที่ (ช่องบรรทัดด้านขวาถูกกำกับด้วย rgtlmt ปกติเท่ากับ wndwid คือความกว้างของบานหน้าต่าง)

● EdiScrn1Ch อ่านคีย์และนำออกแสดงภายในช่องบรรทัด แต่ถ้าเป็นการ delete (ผู้ใช้กดคีย์ Ctrl-D หรือ del) จะลบอักขระตรงเคอร์เซอร์ด้วย DelCh นอกจากนี้ให้เปลี่ยนคีย์คอนโทรลบางตัวคือ Ctrl-D และ Ctrl-S เป็นคีย์หน้าที่พิเศษลูกศรขวาและซ้ายตามลำดับ

● EdiScrn1Row อ่านคีย์และนำออกแสดง ณ บรรทัดที่ row และภายในช่องบรรทัด ซึ่งเริ่มจากคอลัมน์ที่ stt ถึง lmt จนกว่าผู้ใช้จะกดคีย์ Enter หรือคีย์ Esc หรือคีย์คอนโทรลที่ตรงกับอักขระใน ctrlwant หรือคีย์หน้าที่พิเศษใน chfwant ให้ค่าคีย์สุดท้ายใน ch หรือ chf โดยให้ค่า lmt แก่ rgtlmt เป็นขอบเขตของช่องบรรทัดทางขวาแล้วเข้าทำงานเป็นวงรอบ เรียกใช้ EditScrn1Ch นำค่าอักขระมาตรวจว่า

- ถ้า ch ไม่ใช่คีย์คอนโทรลให้ alphakey เป็นจริง
- ถ้า ch เป็น Ctrl-Y ให้ลบอักขระในช่องบรรทัดทั้งหมด
- ถ้า ch เป็น back space และ curx มากกว่า stt ให้เลื่อนเคอร์เซอร์ไปทางซ้าย 1 คอลัมน์ แล้วลบอักขระตรงเคอร์เซอร์ด้วย DelCh
- ถ้า chf เป็นคีย์ลูกศรซ้ายและ curx มากกว่า stt ให้เลื่อนเคอร์เซอร์ไปทางซ้าย 1 คอลัมน์

```
procedure  ReadChChf(var ch,chf : Char);
procedure  DelCh;
procedure  EditScrn1Ch(var ch,chf : Char);
procedure  EditScrn1Row(row,stt,lmt : Integer;
             var ch,chf : Char; ctrlwant,chfwant : ChrSet);
procedure  InScrnStr(var instr : Str79; row,stt,lmt : Integer);
procedure  CursorOn;
procedure  CursorOff;
```

รูปที่ 11

- ถ้า chf เป็นคีย์ลูกศรขวา และ curx น้อยกว่า lmt ให้เลื่อนเคอร์เซอร์ไปทางขวา 1 คอลัมน์

- ถ้า chf เป็นคีย์ Home ให้เลื่อนเคอร์เซอร์ไปที่ต้นช่องหน้าต่าง

- ถ้า chf เป็นคีย์ End ให้เลื่อนเคอร์เซอร์ไปที่หลังอักขระตัวสุดท้ายในช่องบรรทัด

● InScrnStr อ่านอักขระโดยตรงจากจอภาพ ณ โรว์ที่ row และภายในช่องบรรทัดจาก stt ถึง lmt ให้แก่สตริง instr

● CursorOff งดแสดงเคอร์เซอร์บนจอภาพ

● CursorOn ให้แสดงเคอร์เซอร์ (หลังจากงด)

● Fin ยกเลิกหน่วยความจำที่เป็นบัฟเฟอร์ของหน้าต่าง

โปรแกรมย่อย

เป็นโปรแกรมที่สนับสนุนการทำงานของโปรแกรมกิจกรรม และไม่ประสงค์จะให้ผู้เขียนโปรแกรมคนต่อไปนำไปใช้คือ โปรแกรมในรูปที่ 12 ที่เหลือ จึงของดให้คำอธิบายครับ

```
  destructor  Fin;
  procedure   DrawFrame(name : Str79);
  procedure   FrameOnOff(onn : Boolean);
  procedure   SetFullWnd;
  procedure   SetWndCoor;
  procedure   CalBuffSi;
  procedure   DoOpenSub;
  procedure   DoMoveTo(nx1,ny1 : Integer);
  procedure   GetArrowKey(var ch : Char; var step : Integer);
  function    CnvrtHorVal(var row,frm,too : Integer) : Boolean;
  function    CnvrtVerVal(var col,frm,too : Integer) : Boolean;

              END;

  procedure   SetSmplParm(nme : Str49; x1,y1,wid,hi : Byte;
                          var aparm : WndParmT);
```

รูปที่ 12

```
PROGRAM  TestWindow1;

uses  crt,prime,smplwnd;

const
  mainwnd    : WndParmT =
    ( name     : ' MAIN ';
      pale     : (2,2,78,23,1,1);
      horline  : (127,0,0,0,0,0,0,0,0,0);
      verline  : (127,0,0,0,0,0,0,0,0,0));


  tstwnd : WndParmT =
    ( name     : ' TEST WINDOW ';
      pale     : (10,5,40,12,1,1);
      horline  : (6,none,0,0,0,0,0,0,0,0);
      verline  : (22,none,0,0,0,0,0,0,0,0));

  left  : PaleT = (1,1,21,5,1,1);
  right : PaleT = (23,1,18,5,1,1);
  mid   : PaleT = (1,7,40,6,1,1);

type  PtrPaleT = ^PaleT;


var   main,note : SimWnd;  ch,chf : Char;  sx,sy : Integer;
      sttmem : LongInt;  instr : Str79;   opale : PtrPaleT;
```

รูปที่ 13

โปรแกรมนอกสกุล

โดยที่พารามิเตอร์ของหน้าต่างมีรายละเอียดมาก แต่มีงานส่วนหนึ่งที่ไม่ต้องการรายละเอียดเหล่านี้ จึงจัดโปรแกรมย่อย SetSmplParm ไว้ให้ (อยู่ในรูปที่ 12) เป็นโปรแกรมย่อยของยูนิต SMPLWND แต่อยู่นอกสกุล SimWnd เพื่อเตรียมค่าพารามิเตอร์ที่สมบูรณ์ให้แก่ aparm การที่นำมาไว้นอกสกุลเพราะต้องใช้โปรแกรมย่อยนี้ก่อน Init

ตัวอย่างโปรแกรมใช้งาน

สำหรับโปรแกรมของหน้าต่างใหม่ที่สมบูรณ์ กรุณาใช้บริการซอฟต์-เมล์ครับ เพื่อให้เห็นการใช้งานสกุล SimWnd ผมได้เขียนโปรแกรมตัวอย่างไว้ในไฟล์ TSTWND1.PAS เพื่อแสดงถึงการใช้งานที่เพิ่มขึ้นมาโดยจะสร้างหน้าต่างที่มีบานหน้าต่าง 3 บาน ตามรูปที่ 6 และให้ผู้ใช้เลือกเปิดบานหน้าต่างบานใดบานหนึ่งเพื่อป้อนข้อความ กับให้เลือกย้ายหน้าต่างเพื่อแสดงให้เห็นว่าเมื่อย้ายหน้าต่างแล้วจะกลับมายังบานที่เปิดใช้อยู่ กับให้ย่อขยายหน้าต่างได้ เพื่อจะกล่าวถึงการเขียนโปรแกรมของกิจกรรม Show ในสกุลรองต่อไป มีรายละเอียดดังนี้คือ

● ค่าพารามิเตอร์ ได้แสดงการกำหนดค่าพารามิเตอร์ไว้เป็นค่าคงตัว 2 แบบในรูปที่ 13 คือ

- mainwnd เพื่อแสดงค่าพารา-มิเตอร์ของหน้าต่างสูงสุดเท่านั้น (ไม่ได้นำไปใช้) ค่านี้จะไม่มีกรอบหน้าต่าง

- tstwnd คือหน้าต่างที่จะยกมาเป็นตัวอย่างขอให้สังเกตว่าได้ให้ตีเส้นคั่นหน้าต่างในแนวนอนที่โรว์ 6 ของหน้าต่าง และในแนวตั้งที่คอลัมน์ 22 คำว่า none คือค่า 127 เพื่อแสดงว่า

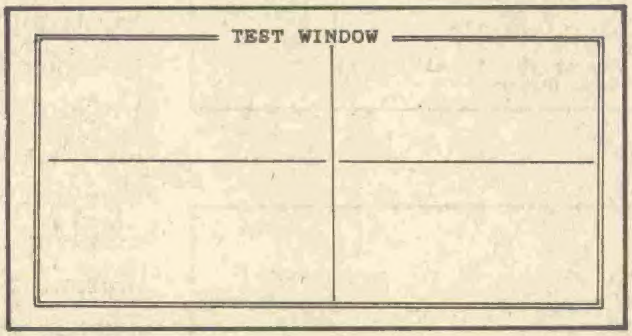
```
procedure  OpenPale(var npale : PaleT);
begin
  note.CloseSubWnd(opale^);
  note.OpenSubWnd(npale);  opale := @npale
end;


procedure  TinyEdit(var ch,chf : Char);
begin
  with  note  do
    begin
    EditScrn1Row(cury,1,wndwid,ch,chf,[],[F1,F2,F3,F4,F5]);
    if  ch = cr  then
     begin  WriteLn;  curx := WhereX;  cury := WhereY  end;
    end;
end;
```

รูปที่ 14

```
procedure   ReFramePale;
var  tem : PtrPaleT;
begin
  tem := opale;
  with  note  do
    begin
    mid[4] := yy2 - yy1 - mid[2];  OpenPale(mid);
    DrawVer(tstwnd.verline[1],0,note.wndhi,space);
    OutChAt(#193,0,tstwnd.verline[1]);
    opale := tem;  OpenSubWnd(opale^);
    end;
end;
```

รูปที่ 15

รูปที่ 16
แสดงการตีเส้นคั่นหน้าต่าง
ในแนวนอนและแนวตั้ง




```
BEGIN
  sttmem := MemAvail;   ClrScr;
  note.Init(tstwnd);  note.Show;
  opale := @tstwnd.pale;  OpenPale(mid);   ReFramePale;

  repeat
    TinyEdit(ch,chf);
    case  chf  of
     F1 : OpenPale(left);
     F2 : OpenPale(right);
     F3 : OpenPale(mid);
     F4 : note.Drag;
     F5 : begin  note.ExpCond;  ReFramePale  end;
    end;  { case }
  until  ch = esc;

   note.Fin;
   WriteLn('.At start mem avail = ',sttmem,' .........');
   WriteLn('......... Mem Avail = ',MemAvail,' .........');
END.
```

รูปที่ 17

ไม่มีเส้นคั่นหน้าต่าง (ต่อไป)

● บานหน้าต่าง ได้กำหนดรายละเอียดของบานหน้าต่างทั้ง 3 บานคือ left (บนซ้าย), right (บนขวา) และ mid (บานล่าง) เป็นค่าคงตัวเช่นกัน โดยที่ในแต่ละบาน เลข 2 ตัวแรกคือ ตำแหน่งมุมบนซ้ายของบานหน้าต่างในหน้าต่าง (จะเห็นว่าบานบนซ้ายขึ้นต้นด้วย 1, 1) ตัวที่ 3 คือความกว้าง ตัวที่ 4 คือความสูง ขอให้เทียบกับเส้นคั่นด้วยครับ

● แบบข้อมูล PtrPaleT ใช้กำหนดตัวแปร opale เพื่อเก็บตำแหน่งเคอร์เซอร์ให้แก่บานหน้าต่างที่จะถูกปิด

● OpenPale ตามรูปที่ 14 เพื่อเปิดใช้บานหน้าต่าง บานที่ผ่านค่ามาให้ npale

● TinyEdit เพื่อให้ผู้ใช้ป้อนข้อความกับจอภาพ

● ReFramPale ดังรูปที่ 15

เมื่อเริ่มต้น Init และให้ Show หรือเมื่อย่อขยายหน้าต่างจากการทำงานของ SimWnd จะเกิดภาพบนจอภาพดังแสดงในรูปที่ 16 ทั้งนี้เพราะผู้เขียนไม่ได้จัดโปรแกรมย่อยสำหรับตีกรอบบานหน้าต่าง และการดำเนินการกับบานหน้าต่างเมื่อขยายหรือย่อหน้าต่าง เพราะเห็นว่าเป็นเรื่องเฉพาะรายและหน้าต่างที่มีบานหน้าต่างหลายบานไม่น่าจะมีการย่อขยายหน้าต่าง

จึงจัดโปรแกรมนี้ไว้ลบเส้นที่เกินมาและทำเป็นกรอบหน้าต่าง ดังรูปที่ 6 แต่ก็ยังมีปัญหาดังจะกล่าวในหัวข้อข้อคิดเห็น

● โปรแกรมหลัก อยู่ในรูปที่ 17 ให้ผู้ใช้ดำเนินการดังกล่าวมา

```
TYPE
  PtrPrompter = ^Prompter;
  Prompter = OBJECT(SimWnd)
    promparm : WndParmT;

  constructor  Init(x1,y1,wid,hi : Byte);
  procedure    Show; virtual;
  procedure    Hide; virtual;
  procedure    PrepPrompt(prompt,quest : Str79; nwid,att : Byte);
  procedure    GetAnsStr(var ans : Str79; len : Integer);
  procedure    GetAnsChCr(var ch : Char; keywant : ChrSet);
  procedure    GetAnsCh(var ch : Char; keywant : ChrSet);

  destructor   Fin;
               END;
```

รูปที่ 18

ข้อคิดเห็น

จากโปรแกรมตัวอย่างนี้จะเห็นว่า SmpWnd ใช้งานได้กับบานหน้าต่างหลายบาน และให้เคลื่อนย้ายได้ด้วย แต่ในการย่อขยายมีปัญหา ปัญหาที่ยังไม่ได้กล่าวถึงคือ เมื่อย่อหรือขยายหน้าต่างจะเป็นการย่อหรือขยายหน้าต่างทั้งหมด แต่บานหน้าต่างยังไม่เปลี่ยนแปลงโดยเฉพาะทางข้าง

หากต้องการให้ย่อขยายหน้าต่างที่มีหลายบานหน้าต่างได้ด้วย ควรทำเป็นสกุลรองและเขียนโปรแกรมของกิจกรรม Show เพิ่ม คือเมื่อเรียกใช้ SimWnd.Show แล้วให้ตรวจดูที่มีการย่อขยายหน้าต่างหรือไม่ โดยตรวจที่ wndwid และ wndhi แล้วจึงตรวจ wx1 และ wy1 (ขณะนี้จะเป็นค่าของหน้าต่างทั้งหน้าต่าง) แล้วปรับแต่งค่าใน left, right และ mid ให้สอดคล้องกัน

พรอมพ์เตอร์

เพื่อการใช้งานในโครงงานใหม่และเก่าผมได้กำหนดสกุล Prompter และเก็บไว้ในยูนิต SMPLWND ด้วย พรอมพ์เตอร์คือหน้าต่างเล็ก ๆ ที่มีคำถามให้ผู้ใช้ตอบเป็นสตริงหรืออักขระดังที่พบบ่อย ๆ ขณะที่ใช้งานเทอร์โบปาสคาล เพื่อท่านผู้อ่านจะนำไปใช้ก่อน ขอแสดงวิธีใช้ดังนี้ครับ (ตามรูปที่ 8)

- Init เตรียมหน้าต่างด้วยจุดมุมบนซ้าย (x1, y1) และความกว้าง (wid) ตามต้องการกับความสูง (hi) ปกติคือ 1

- PrepPrompt ในการใช้งานพรอมพ์เตอร์ ปกติจะกำหนดเป็นอินสแตนซ์ (ตัวแปร) ไว้ในสกุลอื่น ซึ่งถึงแม้จะมีข้อความถามตอบหลายแบบก็ไม่จำเป็นต้องกำหนดอินสแตนซ์ไว้หลายตัว เมื่อจะใช้คำถามใด ให้ใช้ PrepPrompt โดยที่ promt คือ ข้อความที่จะแสดงตรงกรอบเช่น Verify ส่วน quest คือคำถามหรือคำอธิบาย เช่น Qverwrite TEM.PAS? (Y/N) สำหรับ nwid คือค่าความกว้างของหน้าต่างใหม่ที่เหมาะสมกับ quest และ ans หรือ ch หากไม่เปลี่ยนให้เป็น 0 ส่วน att คือแอตทริบิวต์ของคำถามที่ต้องการเน้นในแบบใด เสร็จแล้วต้องใช้กิจกรรมอ่านคำตอบอันใดอันหนึ่งต่อไปนี้คือ

- GetAnsStr เมื่อต้องการคำตอบเป็นสตริงให้แก่ astr ยาวไม่เกิน len อักขระ

- GetAnsCh เมื่อต้องการคำตอบเป็นอักขระที่ระบุไว้ใน keywant ได้ค่าคีย์ใน ch

- GetAnsChCr เช่นเดียวกับ GetAnsCh แต่ต้องการให้ผู้ใช้กดคีย์ Enter ด้วย

เพื่อความเข้าใจเพิ่มขอให้ดูและทดลองรันโปรแกรมในไฟล์ TSTPRMPT.PAS หวังว่าคุณผู้อ่านคงได้ใช้ประโยชน์สกุลทั้งสองนี้ ส่วนผมอาจต้องว่างเว้นไประยะหนึ่งครับ